Butterworth Low-Pass Filters

In this article, we describe the commonly-used, n^{th}-order Butterworth low-pass filter. First, we show how to use known design specifications to determine filter order and 3dB cut-off frequency. Then, we show how to determine filter poles and the filter transfer function. Along the way, we describe the use of common *Matlab Signal Processing Toolbox* functions that are useful in designing Butterworth low-pass filters.

The squared magnitude function for an n^{th}-order Butterworth low-pass filter is

$$\left|H_a(j\Omega)\right|^2 = H_a(j\Omega)H_a^*(j\Omega) = \frac{1}{1+(j\Omega/j\Omega_c)^{2n}}, \tag{1-1}$$

where constant Ω_c is the 3dB cut-off frequency. Magnitude $|H_a(j\Omega|$ is depicted by Figure 1.

It is easy to show that the first 2n-1 derivatives of $|H_a(j\Omega|^2$ at $\Omega = 0$ are equal to zero. For this reason, we say that the Butterworth response is *maximally flat* at $\Omega = 0$. Furthermore,

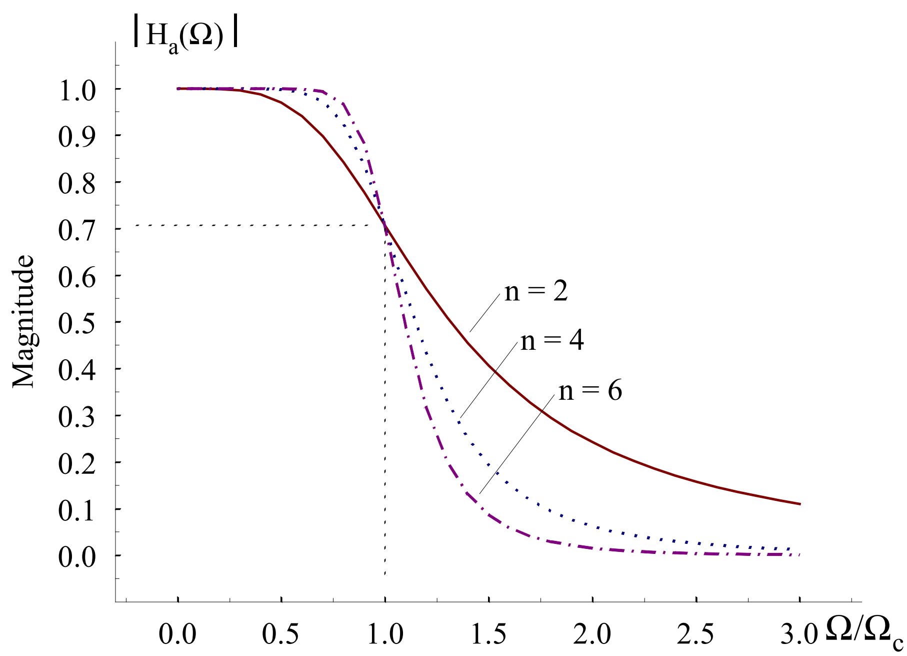

Figure 1: Magnitude response of an ideal n^{th}-order Butterworth filter.

the derivative of the magnitude response is always negative for positive Ω, the magnitude response is monotonically decreasing with Ω. For $\Omega >> \Omega_c$, the magnitude response can be approximated by

$$\left|H_a(j\Omega)\right|^2 \approx \frac{1}{(\Omega/\Omega_c)^{2n}}. \tag{1-2}$$

Butterworth Filter Design Procedure

We start with specifications for the filter. Usually, the specifications are given as

$$\begin{aligned}
&1)\ \Omega_p \text{ the pass-band edge,}\\
&2)\ \Omega_s \text{ the stop-band edge,}\\
&3)\ \left|H_a(j\Omega_p)\right| = 1\Big/\sqrt{1+\varepsilon^2}, \text{ the maximum pass-band attenuation or ripple,}\\
&4)\ \left|H_a(j\Omega_s)\right| = 1\big/A, \text{ the minimum stop-band attenuation or ripple.}
\end{aligned} \tag{1-3}$$

These four pieces of known information must be used to compute filter order n, 3dB cut-off frequency Ω_c and filter transfer function $H_a(j\Omega)$. For example, this is the approach used by the Butterworth design functions in the Matlab Signal Processing Toolbox.

The third specification in (1-3) can be used to write

$$\left|H_a(j\Omega_p)\right| = 1\Big/\sqrt{1+(\Omega_p/\Omega_c)^{2n}} = 1\Big/\sqrt{1+\varepsilon^2}, \tag{1-4}$$

a result that leads to $(\Omega_p/\Omega_c)^{2n} = \varepsilon^2$ and the requirement

$$n = \frac{\log\varepsilon}{\log\Omega_p - \log\Omega_c}, \tag{1-5}$$

a single equation in the unknowns Ω_c and n. Equation (1-5) can be used to write

$$\log\Omega_c = \log\Omega_p - \frac{1}{n}\log\varepsilon. \tag{1-6}$$

We need a second equation in the unknowns Ω_c and n. This second equation is obtained from the fourth of (1-3), a specification on the stop band. At stop-band edge, we have

$$\left|H_a(j\Omega_s)\right|^2 = \frac{1}{1+(\Omega_s/\Omega_c)^{2n}} = \frac{1}{A^2}, \tag{1-7}$$

a result that leads to $(\Omega_p/\Omega_c)^{2n} = A^2 - 1$ or

$$n = \frac{\log\sqrt{A^2-1}}{\log\Omega_p - \log\Omega_c}. \tag{1-8}$$

This second equation in the unknowns Ω_c and n can be used to write

$$\log\Omega_c = \log\Omega_s - \frac{1}{n}\log\sqrt{A^2-1}. \tag{1-9}$$

With (1-5) and (1-8), we have two equations in the unknowns Ω_c and n. To solve these, substitute (1-9) into (1-5) and obtain

$$n = \frac{\log \varepsilon}{\log \Omega_p - \log \Omega_s + \frac{1}{n}\log\sqrt{A^2 - 1}}, \tag{1-10}$$

a result that can be solved for

$$n = \frac{\log\left(\varepsilon \Big/ \sqrt{A^2 - 1} \right)}{\log\left(\Omega_p \Big/ \Omega_s \right)}, \tag{1-11}$$

a simple formula for the necessary filter order n. Of course, in the likely event that (1-11) yields a fractional value, it must be rounded up to the next integer value, so that n is a positive integer.

Using the just-determined integer value of n, we can solve for Ω_c by using either (1-6) *or* (1-9). If (1-6) is used, we will meet the third specification (the pass-band specification) in (1-3) and possibly exceed (because we may have rounded up to obtain an integer for n) the fourth specification (the stop-band specification). On the other hand, if (1-9) is used to calculate Ω_c, we will meet the stop-band specification and possibly exceed the pass-band specification.

The Matlab Signal Processing Toolbox function `Buttord` uses the just-outlined approach. Using the Matlab definitions

`Wp` for the pass-band edge **in radians/second**

`Ws` for the stop-band edge **in radians/second**

`Rp` for the maximum pass-band ripple (the attenuation, **in dB**, at `Wp`)

`Rs` for the minimum stop-band ripple (the attenuation, **in dB**, at `Ws`), we compute filter-order n and 3dB down frequency Wc with the command-line statement

```
[n,Wc] = buttord(Wp,Ws,Rp,Rs,'s')
```
(1-12)

Matlab rounds up (1-11) to determine n and uses (1-9) to compute Wc. So, buttord selects n and Wc to meet the stop-band specification and (possibly) exceed the pass-band specification.

Filter Transfer Function $H_a(j\Omega)$

Next, we must obtain the transfer function $H_a(j\Omega)$ for the just-computed values of n and Ω_c. Note that we require impulse response $h_a(t)$ to be real-valued and causal. This requirement leads to

$$H_a{}^*(j\Omega) = \left[\int_0^\infty h_a(t)e^{-j\Omega t}dt\right]^* = \int_0^\infty h_a(t)e^{j\Omega t}dt = H_a(-j\Omega)\,, \tag{1-13}$$

so that

$$H_a(j\Omega)H_a(-j\Omega) = \frac{1}{1+(j\Omega/j\Omega_c)^{2n}}\,. \tag{1-14}$$

Note that there is no real value of Ω for which $H_a(\Omega) = \infty$. That is, $H_a(j\Omega)$ has no poles on the $j\Omega$-axis of the complex s-plane.

We require the filter to be causal and stable. Causality requires $h_a(t) = 0$, $t < 0$. Causality and stability require

$$\int_0^\infty \left|h_a(t)\right| dt < \infty . \tag{1-15}$$

Causality and stability requires that the time-invariant filter have all of its poles in the left-half of the complex s-plane (no $j\Omega$-axis, or right-half-plane poles). The region of convergence for $H_a(s)$ is of the form $\mathrm{Re}(s) > \sigma$ for some $\sigma < 0$, see Figure 2. Hence, $H_a(s)$ can be obtained from $H_a(j\Omega)$, the Fourier transform of the impulse response, by replacing s with $j\Omega$. In (1-14), replace $j\Omega$ with s to obtain

$$H_a(s)H_a(-s) = \frac{1}{1+(s/j\Omega_c)^{2n}} . \tag{1-16}$$

As can be seen from inspection of (1-16), the poles of $H_a(s)H_a(-s)$ are the roots of $1 + (s/j\Omega_c) = 0$. That is, each pole must be one of the numbers

$$s_p = j\Omega_c(-1)^{1/2n} . \tag{1-17}$$

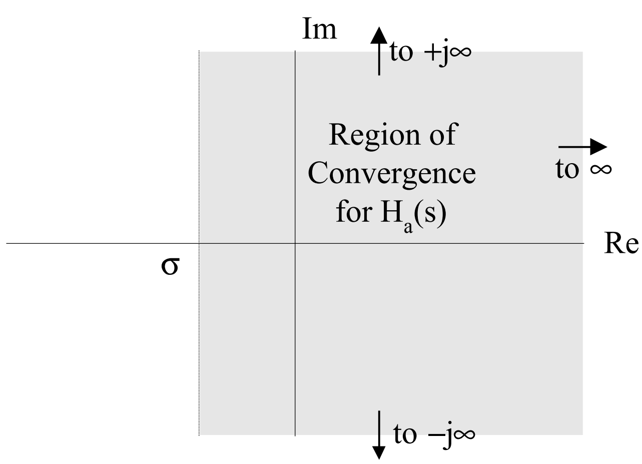

Figure 2: Region of convergence for transform $H_a(s)$, the s-domain transfer function of a n^{th}-order Butterworth filter. Value $\sigma < 0$ depends on cut-off frequency Ω_c and filter order n. All poles of $H_a(s)$ must have a real part that is less than, or equal to, σ.

There are 2n distinct values of s_p; they are found by multiplying the 2n roots of -1 by the complex constant $j\Omega_c$.

The 2n roots of -1 are obtained easily. Complex variable z is a 2n root of -1 if

$$z^{2n} = -1. \tag{1-18}$$

Clearly, z must have unity magnitude and phase $\pi/2n$, modulo 2π. Hence, the 2n roots of -1 form the set

$$\left[1\measuredangle\left\{\tfrac{\pi}{2n}+\tfrac{\pi}{n}k\right\},\ \ k = 0, 1, 2, \cdots, 2n\text{-}1\right]. \tag{1-19}$$

Multiply these roots of -1 by $j\Omega_c$ to obtain the poles of $H_a(s)H_a(-s)$. This produces

$$p_k = \Omega_c\measuredangle\left\{\tfrac{\pi}{2}+\tfrac{\pi}{2n}+\tfrac{\pi}{n}k\right\},\ \ k = 0, 1, 2, \cdots, 2n\text{-}1. \tag{1-20}$$

as the poles of $H_a(s)H_a(-s)$. Notice that p_0, p_1, …, p_{n-1} are in the left-half of the complex plane, while p_n, …, p_{2n-1} are in the right-half plane. In the complex plane, these poles are on a circle (called the *Butterworth Circle*) of radius Ω_c, and they are spaced π/n radians apart in angle. The poles given by (1-20) are

1) symmetric with respect to both axes,

2) never fall on the $j\Omega$ axis,

3) a pair falls on the real axis for n odd but not for n even,

4) lie on the *Butterworth circle* of radius Ω_c where they are spaced π/n radians apart in angle, and

5) half of their number are in the right-half plane and half are in the left-half-plane.

Using (1-20), we can write

$$H_a(s)H_a(-s) = \frac{1}{1+(s/j\Omega_c)^{2n}} = \frac{p_0 p_1 \cdots p_{2n-1}}{(s-p_0)(s-p_1)\cdots(s-p_{2n-1})}. \tag{1-21}$$

Since the poles of $H_a(s)$ are in the left-half plane, we factor (1-21) to produce

$$H_a(s) = \frac{(-1)^n p_0 p_1 \cdots p_{n-1}}{(s-p_0)(s-p_1)\cdots(s-p_{n-1})}, \tag{1-22}$$

the transfer function of the n^{th}-order Butterworth filter. So far, we have required a unity DC gain for the filter (*i.e.*, $H_a(0) = 1$). However, any DC gain can be obtained by simply multiplying (1-22) by the correct constant.

Example: Determine the transfer function for a unity-DC-gain, third-order Butterworth filter with a cut-off frequency of $\Omega_c = 1$ radian/second. The *Butterworth circle*, and 6 poles of $H_a(s)H_a(-s)$, are depicted by Figure 3. The poles of $H_a(s)$ are given by (1-20); these numbers are

$$\begin{aligned} &p_0 = 1\measuredangle\{\pi/2+\pi/6\} = 1\measuredangle\tfrac{2\pi}{3}, \quad p_1 = 1\measuredangle\{\pi/2+\pi/6+\pi/3\} = 1\measuredangle\pi, \\ &p_2 = 1\measuredangle\{\pi/2+\pi/6+2\pi/3\} = 1\measuredangle-\tfrac{2\pi}{3}. \end{aligned} \tag{1-23}$$

Finally, the s-domain transfer function is given by

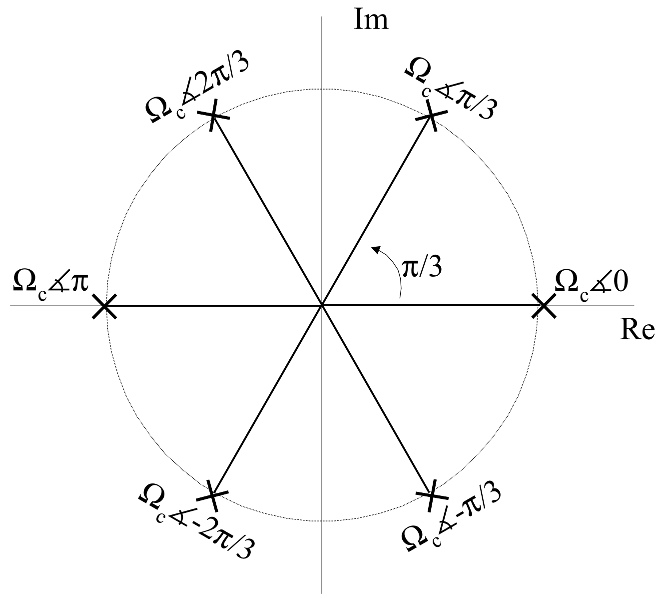

Figure 3: S-plane Butterworth circle of radius $\Omega_c = 1$ for a third-order filter.

$$H_a(s) = \frac{1}{(s - 1\measuredangle\frac{2\pi}{3})(s - 1\measuredangle\pi)(s - 1\measuredangle -\frac{2\pi}{3})} = \frac{1}{s^3 + 2s^2 + 2s + 1} \tag{1-24}$$

Example: The Matlab Signal Processing Toolbox has several powerful functions that are useful for designing Butterworth (and other types of) filters. For example, the code

```
N = 3;
W = 1;
[num,den] = butter(N,W,'s')
```

will design the 3rd-order Butterworth filter that is discussed in the previous example. N is the filter order. W is the 3dB cut-off frequency, `num` is a 1×3 vector of numerator coefficients, and `dom` is a 1×3 vector of denominator coefficients (the coefficient vectors are ordered highest to lowest power of s). To try out "butter", one can type `[num,den]=butter(3,1,'s') at` the Matlab command prompt to obtain

```
>> [num,den]=butter(3,1,'s')
```

```
num =
         0         0         0    1.0000
den =
    1.0000    2.0000    2.0000    1.0000
>>
```

Notice that Matlab returns numerator and denominator polynomial coefficients that agree with the right-hand side of (1-24).